

YDB Topic Service: надёжная и масштабируемая очередь сообщений

Ильдар Хисамбеев,
разработчик систем
поставки данных,
Яндекс



Яндекс



YDB Topics

Это масштабируемый сервис для передачи упорядоченных потоков данных

Сообщество YDB в Telegram
https://t.me/ydb_ru

@YDB_RU

- Распределённая очередь сообщений
- Заменил решение на основе Kafka
- Параметры основного кластера в Яндексе:

  Используют ~1000 команд

  ~70 Гбайт/с на запись

  ~350 К партиций в ~10 К топиках

- Доступен с API Yandex Data Streams
- **Вышел в опенсорс в составе YDB**

Содержание

1. Мотивация
2. Модель очереди на основе лога
3. Архитектура решения
4. Использование YDB Topics

1. Мотивация
2. Модель очереди на основе лога
3. Архитектура решения
4. Использование YDB Topics

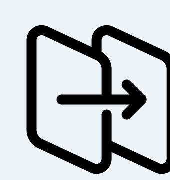
Мотивация

- Задачи для YDB Topics
- Требования к решению
- Зачем строить своё решение

Задачи для YDB Topics

Поставка данных в Data Warehouse

Поставка больших объёмов логов в системы аналитики



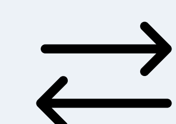
Streaming platform

Аналитика в реальном времени



Async messaging

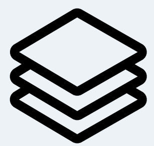
Классический асинхронный обмен сообщениями между сервисами



Infrastructure component

Инфраструктурный компонент

Пример: Change Data Capture YDB-таблиц



Требуемые свойства

1

Надёжность

2

Масштабируемость

3

Доступность

4

Производительность

Требуемые свойства

1

Надёжность

Сообщения не теряются

Настраиваемый FIFO-порядок

Поддержка exactly-once-семантики доставки при записи

2

Масштабируемость

3

Доступность

4

Производительность

Требуемые свойства

1

Надёжность

Сообщения не теряются

Настраиваемый FIFO-порядок

Поддержка exactly-once-семантики доставки при записи

2

Масштабируемость

По throughput на запись

По числу партиций

По числу изолированных пользователей

Издержки на эксплуатацию — O(1)

3

Доступность

4

Производительность

Требуемые свойства

1

Надёжность

Сообщения не теряются

Настраиваемый FIFO-порядок

Поддержка exactly-once-семантики доставки при записи

2

Масштабируемость

По throughput на запись

По числу партиций

По числу изолированных пользователей

Издержки на эксплуатацию — O(1)

3

Доступность

Отказы дисков, хостов — ежедневно, ДЦ — авария или регулярные учения

Переживать автоматически и без деградации по latency

Возможная георaспределённость

4

Производительность

Требуемые свойства

1

Надёжность

Сообщения не теряются

Настраиваемый FIFO-порядок

Поддержка exactly-once-семантики доставки при записи

2

Масштабируемость

По throughput на запись

По числу партиций

По числу изолированных пользователей

Издержки на эксплуатацию — O(1)

3

Доступность

Отказы дисков, хостов — ежедневно, ДЦ — авария или регулярные учения

Переживать автоматически и без деградации по latency

Возможная georаспределённость

4

Производительность

Throughput — десятки Гбайт/с

Latency < 1 с от отправки писателем до получения читателем

Зачем строить своё решение

Решение Apache Kafka®

Почему не подошло

ZooKeeper

Maintainability

Недостаточная изоляция клиентов

Своё решение

Бонусы

Оптимизации под свои задачи

Переиспользование компонентов

Развитие в нужных направлениях

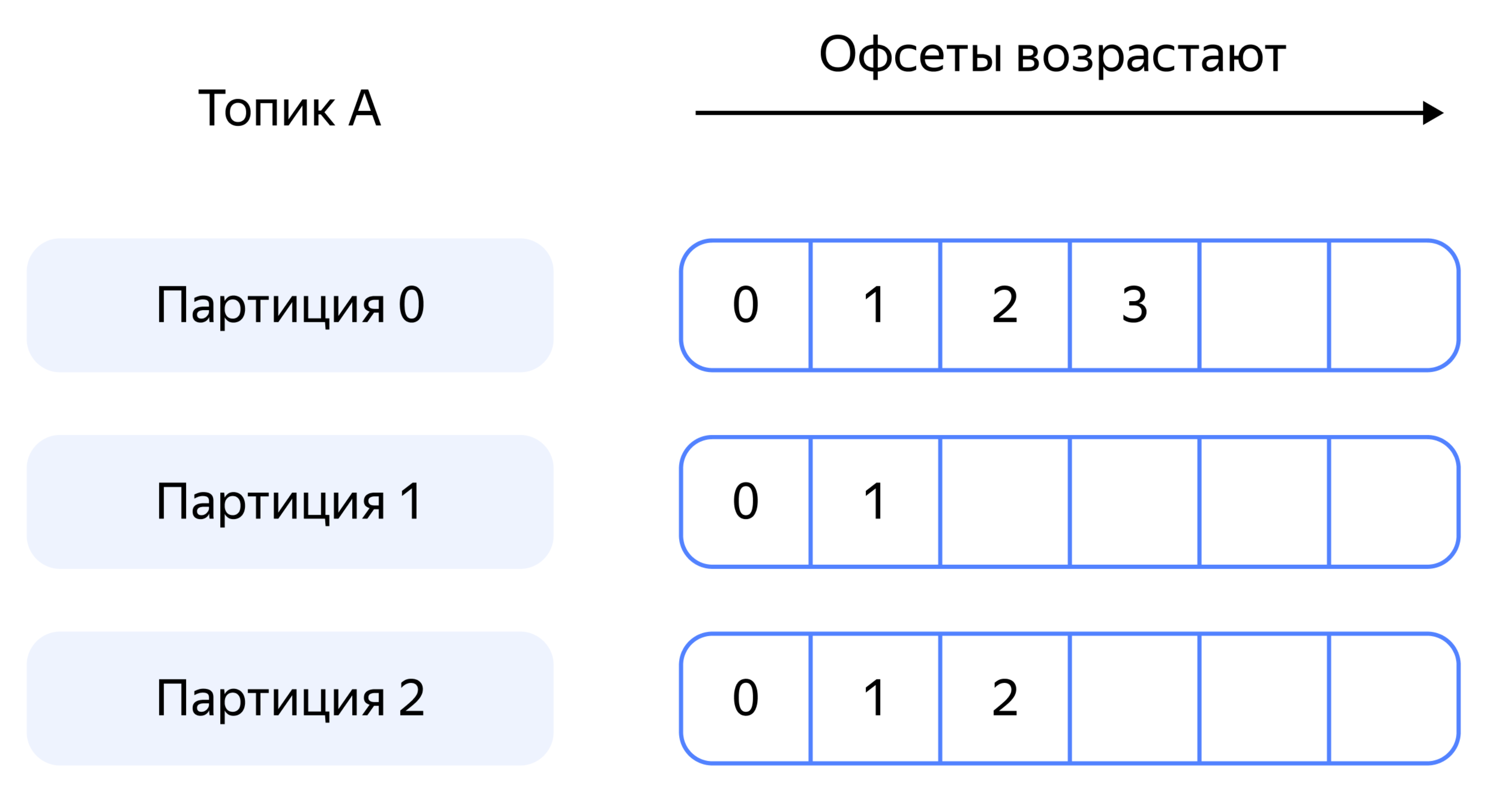
1. Мотивация
2. Модель очереди на основе лога
3. Архитектура решения
4. Использование YDB Topics

Модель очереди на основе лога

- Сущности YDB Topics
- Протоколы передачи сообщений
- Отказы и exactly-once семантика

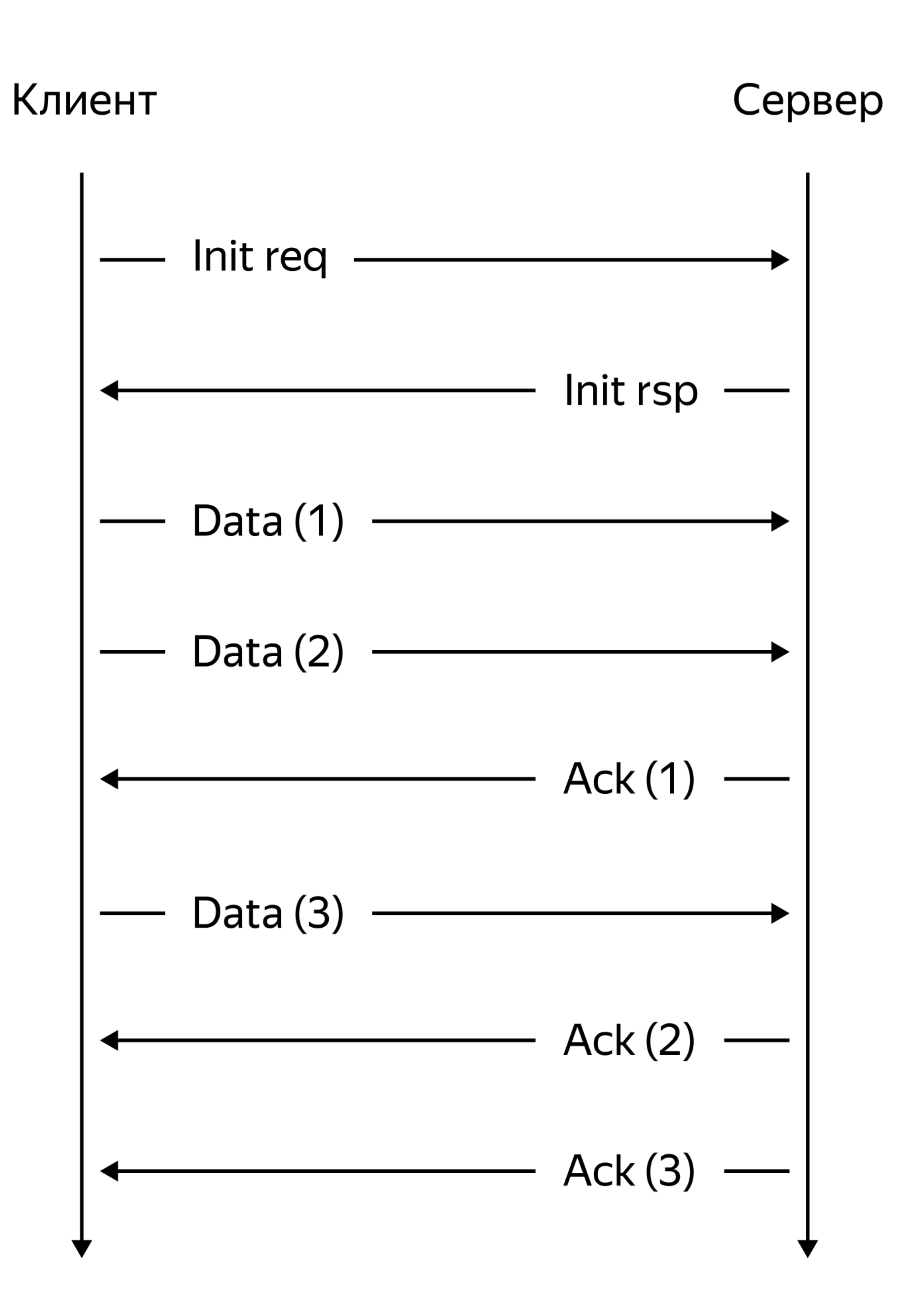
Сущности YDB Topics

- Пользовательские данные сгруппированы по топикам
- Топик разделён на партиции
- Одна партиция — это распределённый лог сообщений
- Сообщение = тело + метаданные
- Номер сообщения в партиции — офсет



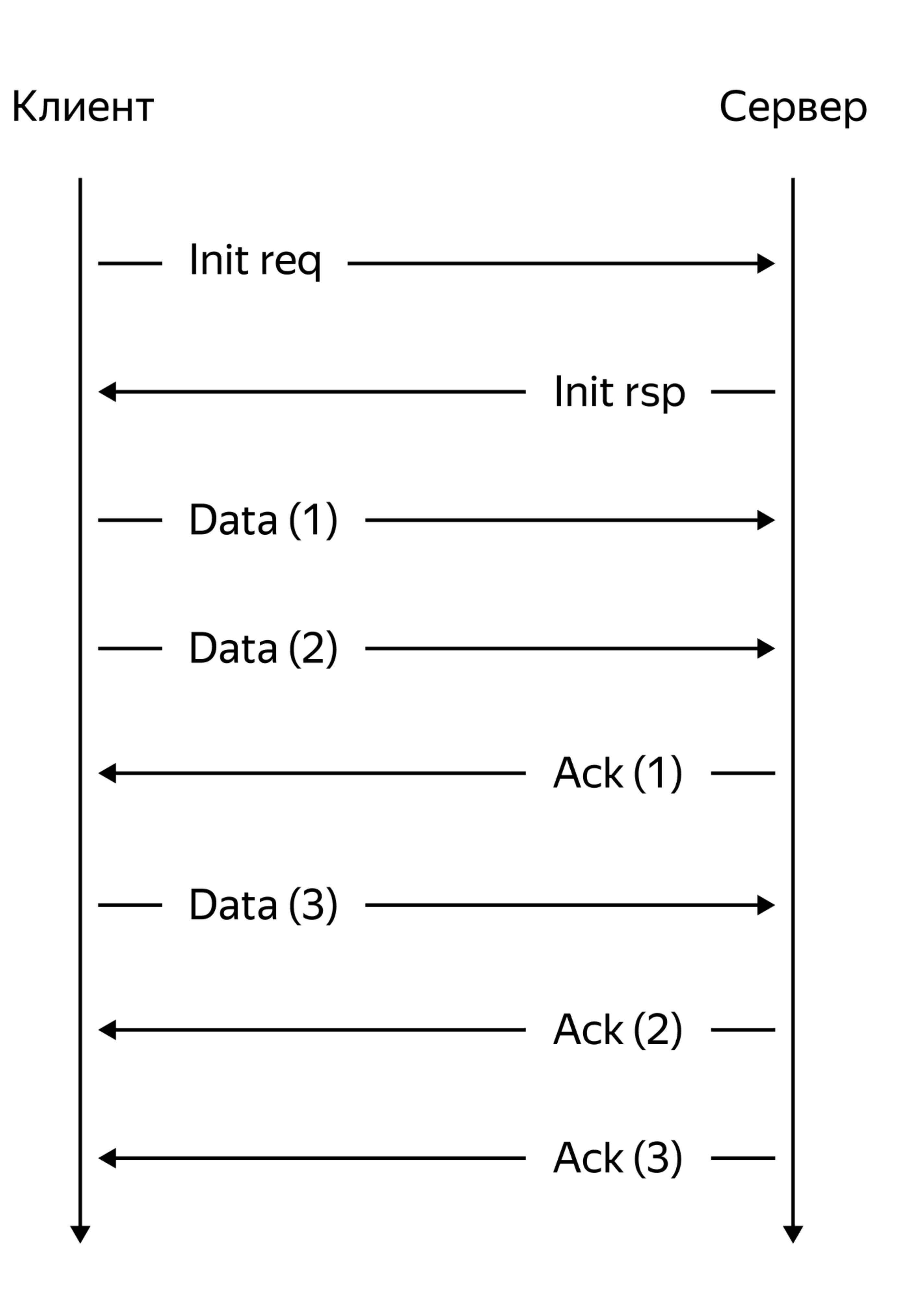
Запись

- Сессия записи — двусторонний потоковый RPC
- Пишем в одну партицию одного топика
- Message_group_id — ключ партиционирования
- Сервер возвращает подтверждения записи

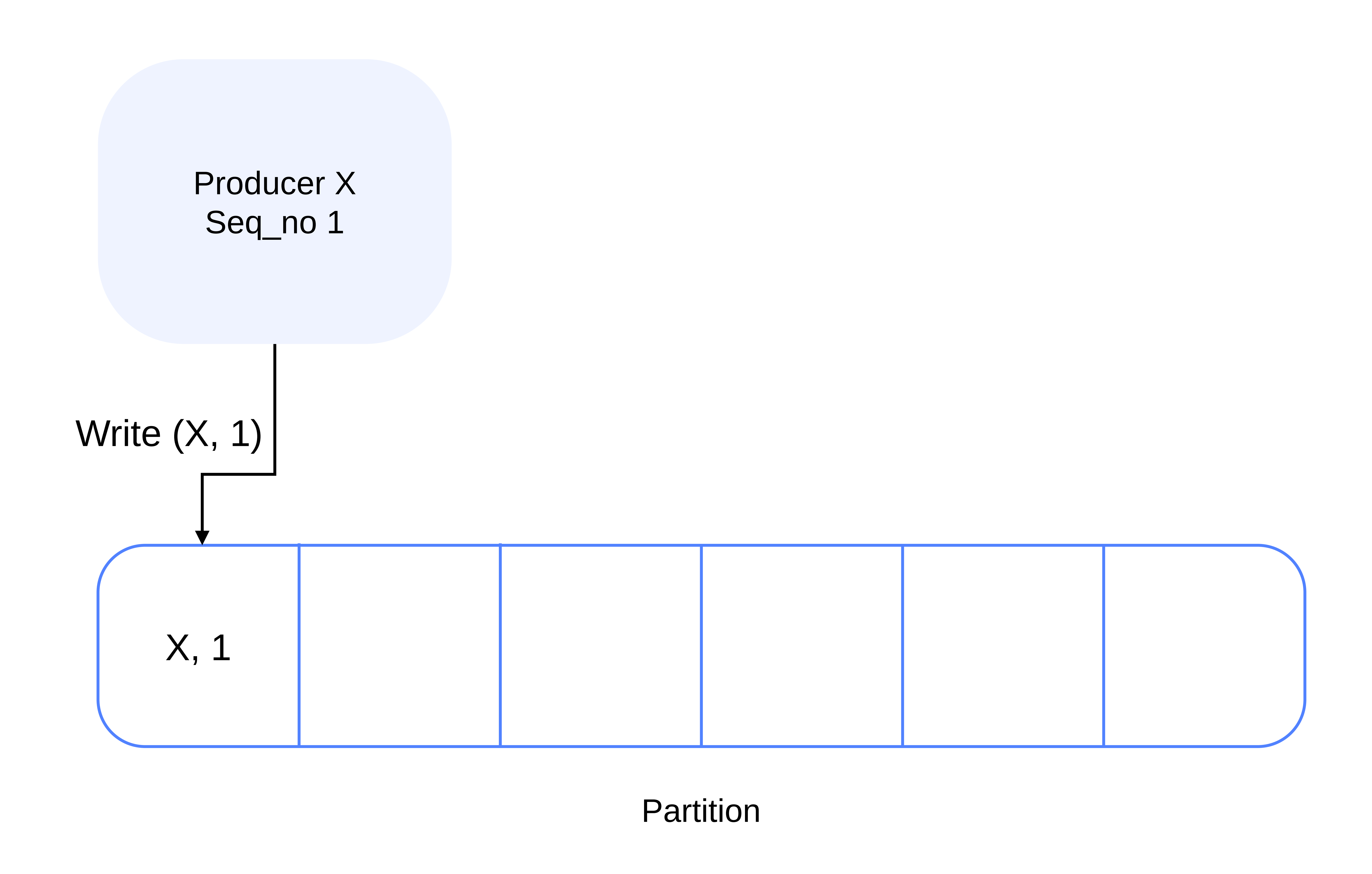


Запись. Дедупликация

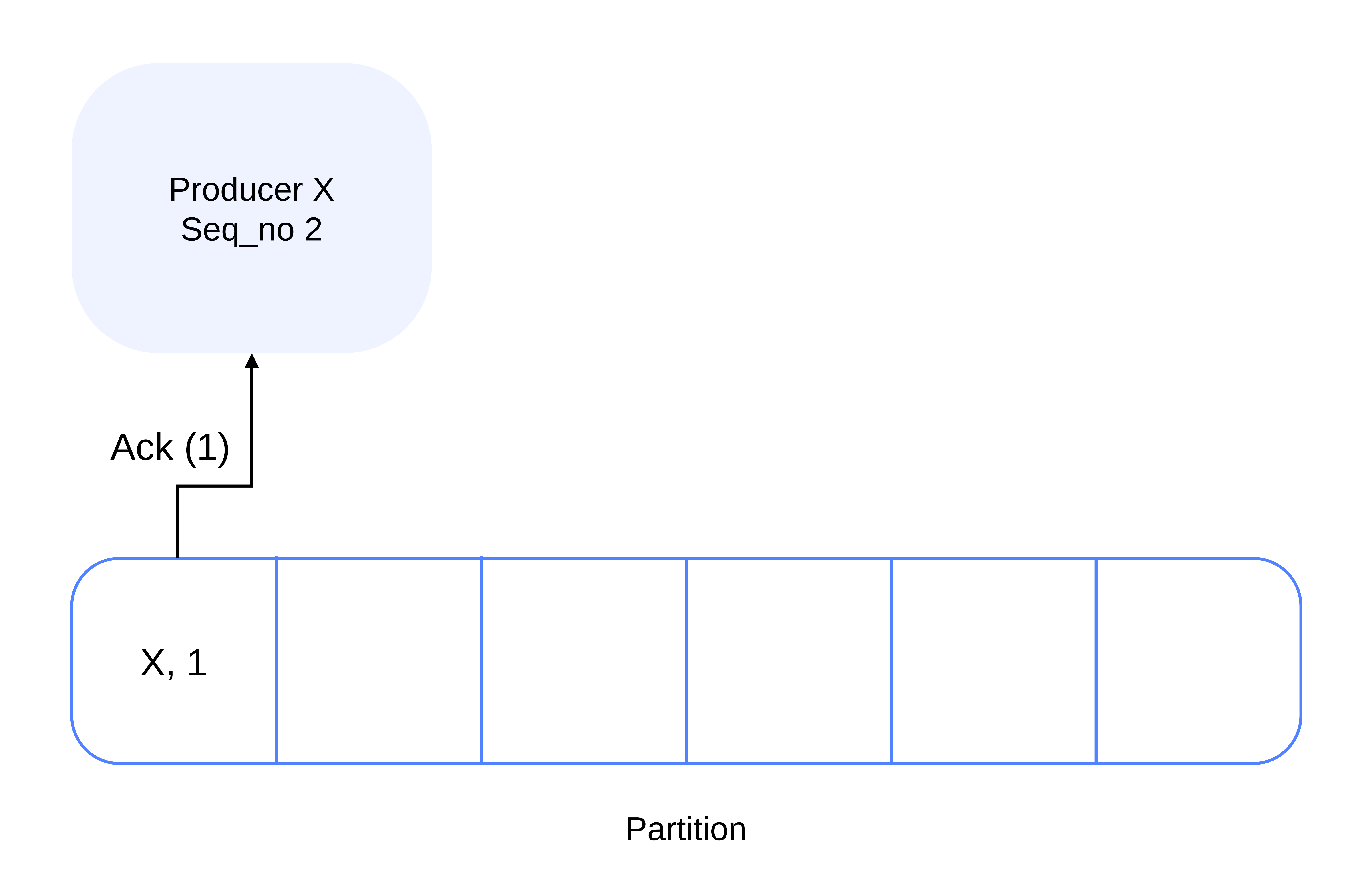
- При сбоях возможна переотправка сообщений
- Добавим к сессии producer_id, к сообщениям – seq_no
- По паре (producer_id, seq_no) узнаём дубли



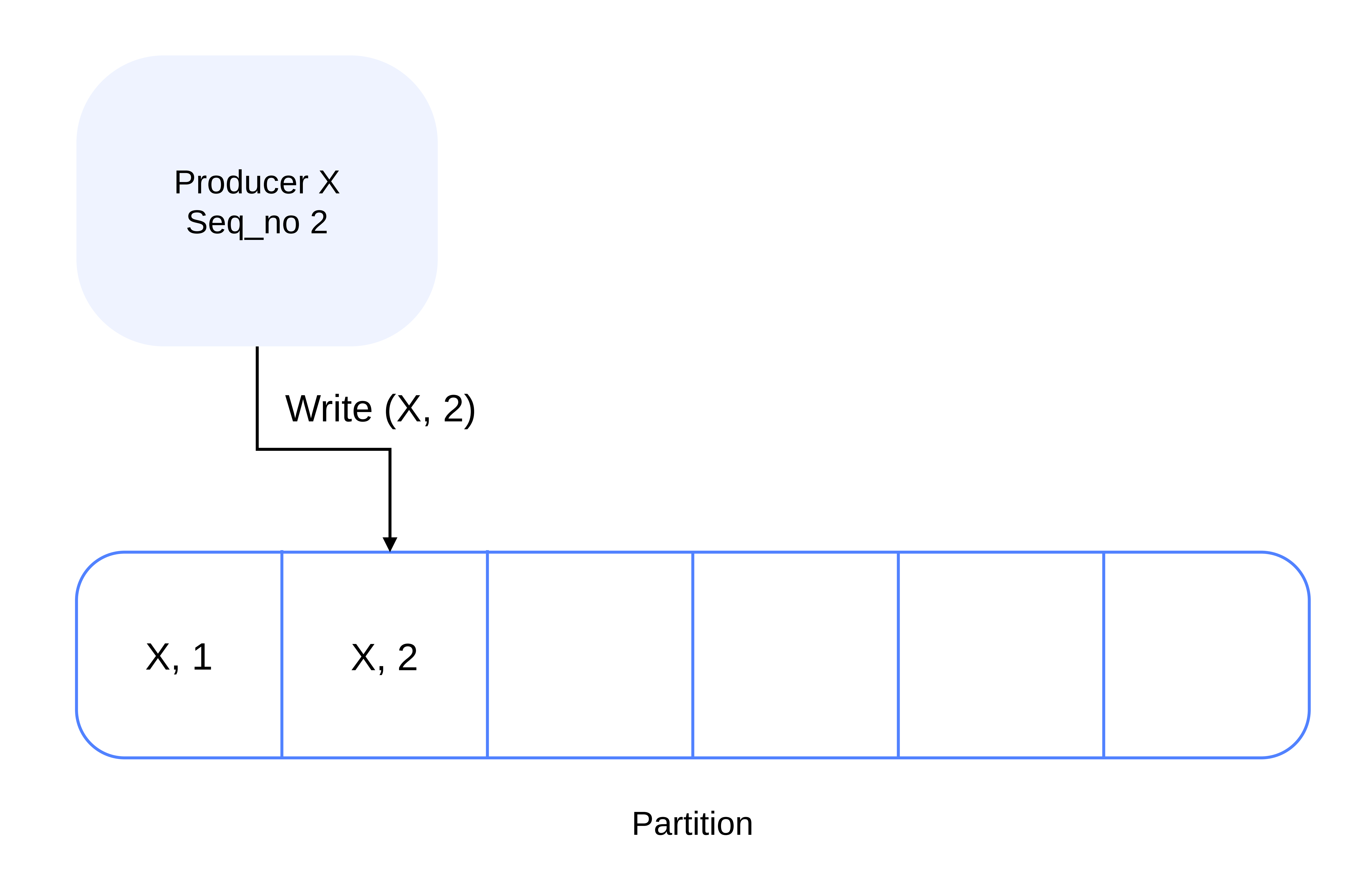
Запись. Дедупликация



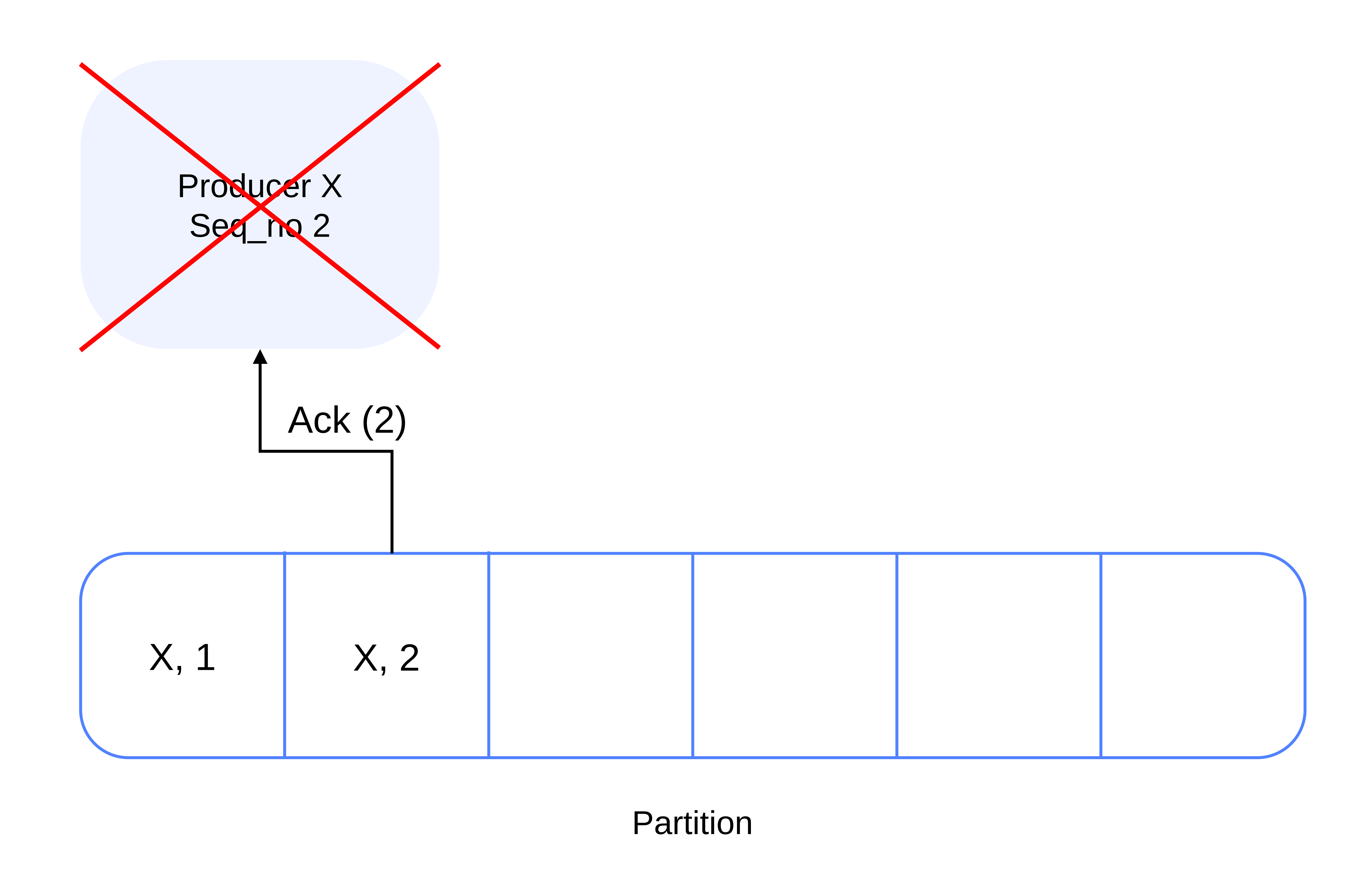
Запись. Дедупликация



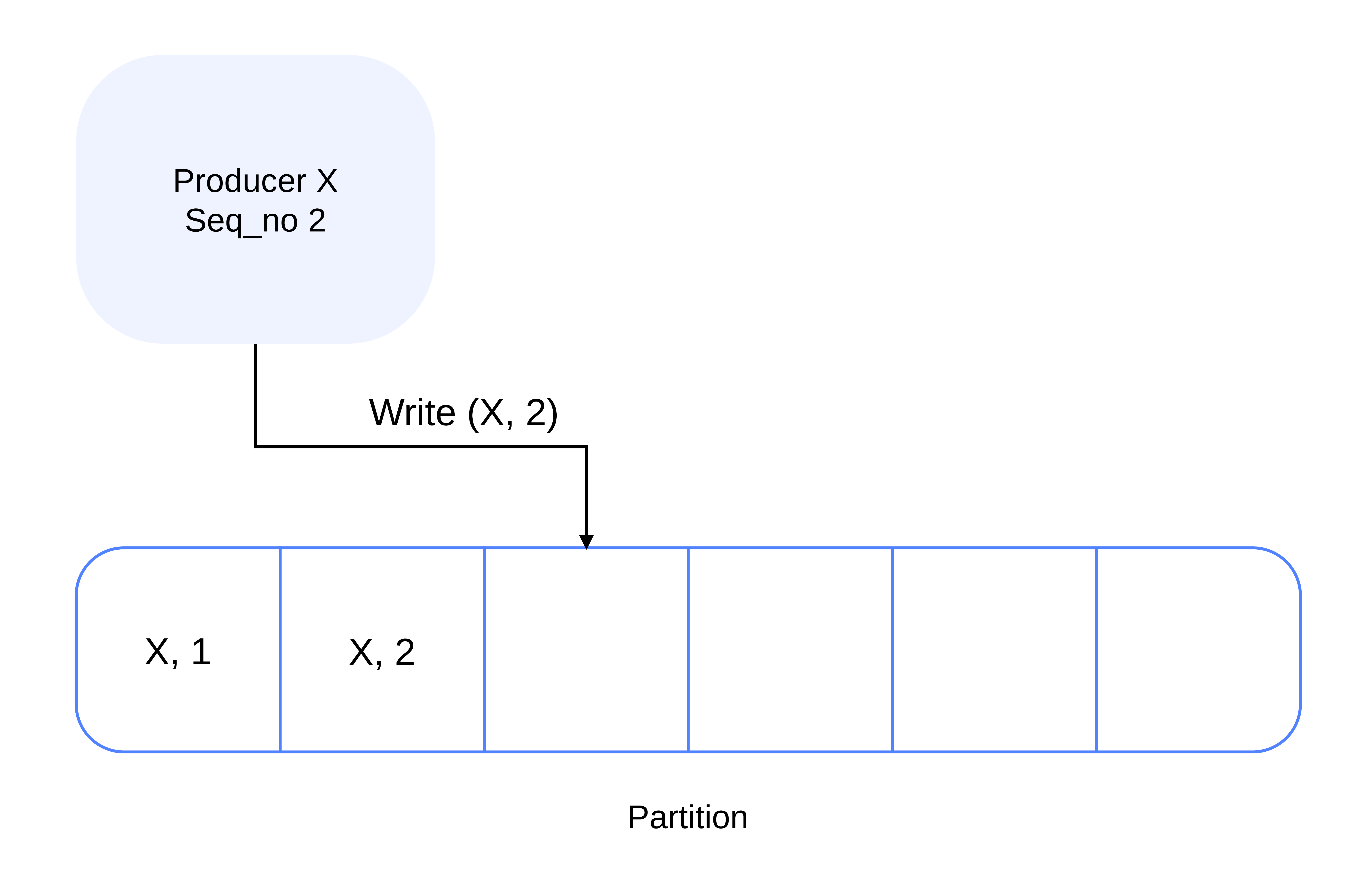
Запись. Дедупликация



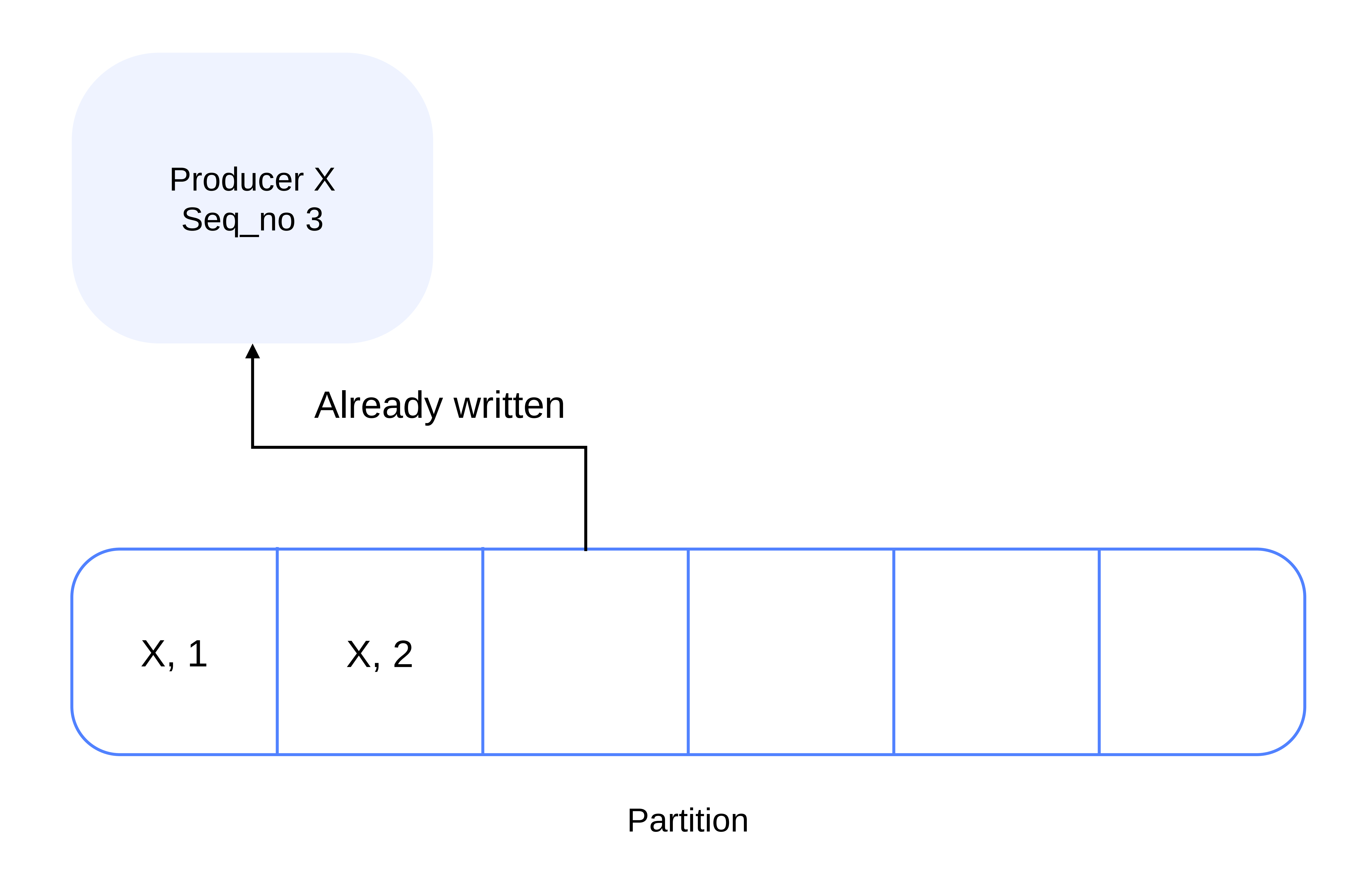
Запись. Дедупликация



Запись. Дедупликация

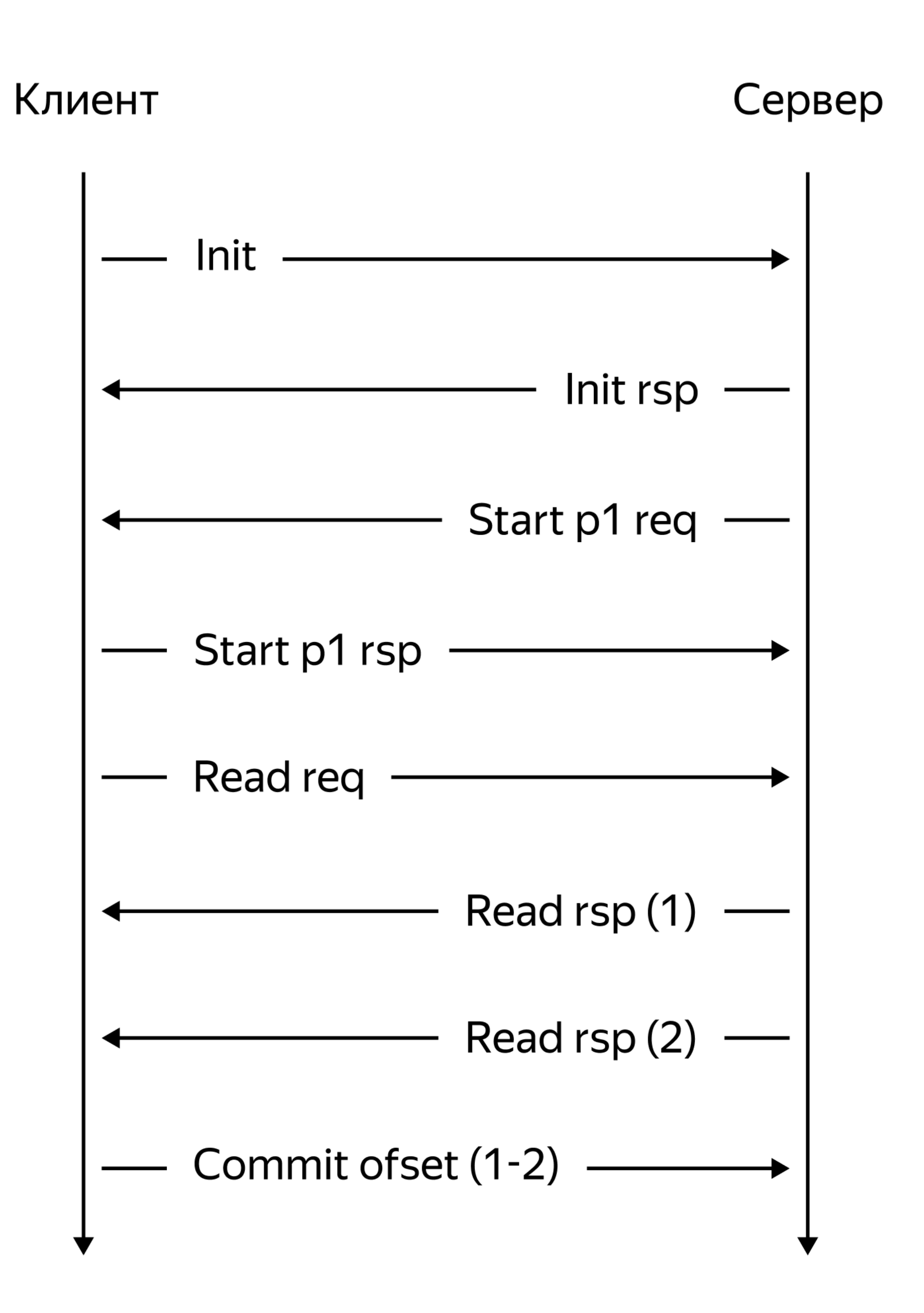


Запись. Дедупликация



Чтение

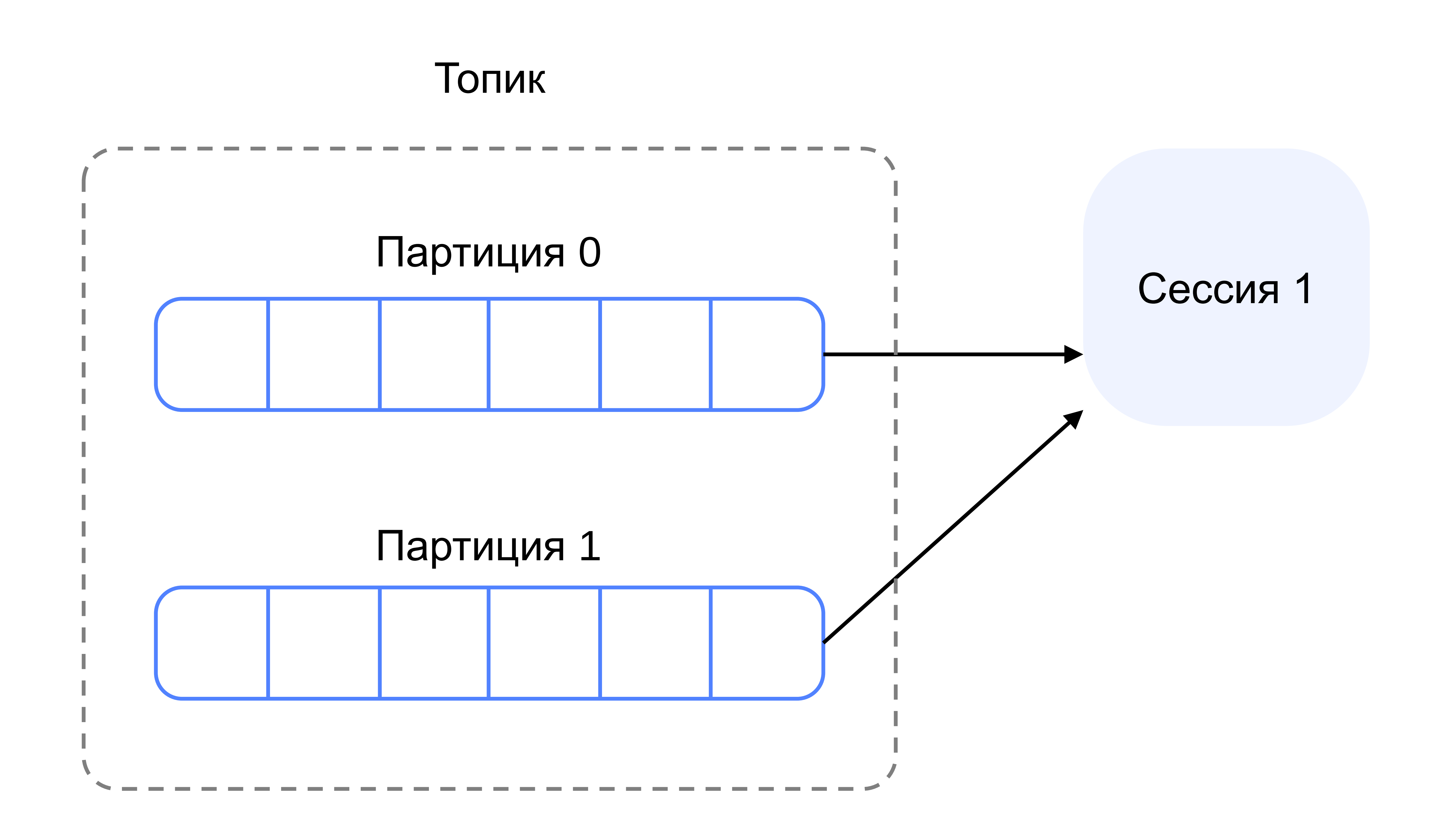
- Сессия чтения — двусторонний потоковый RPC
- Клиент-читатель задаёт consumer_id
- С различными consumer_id читаем независимо
- Дедупликация возможна на стороне клиента по паре «партиция – офсет»
- Клиент-серверная балансировка



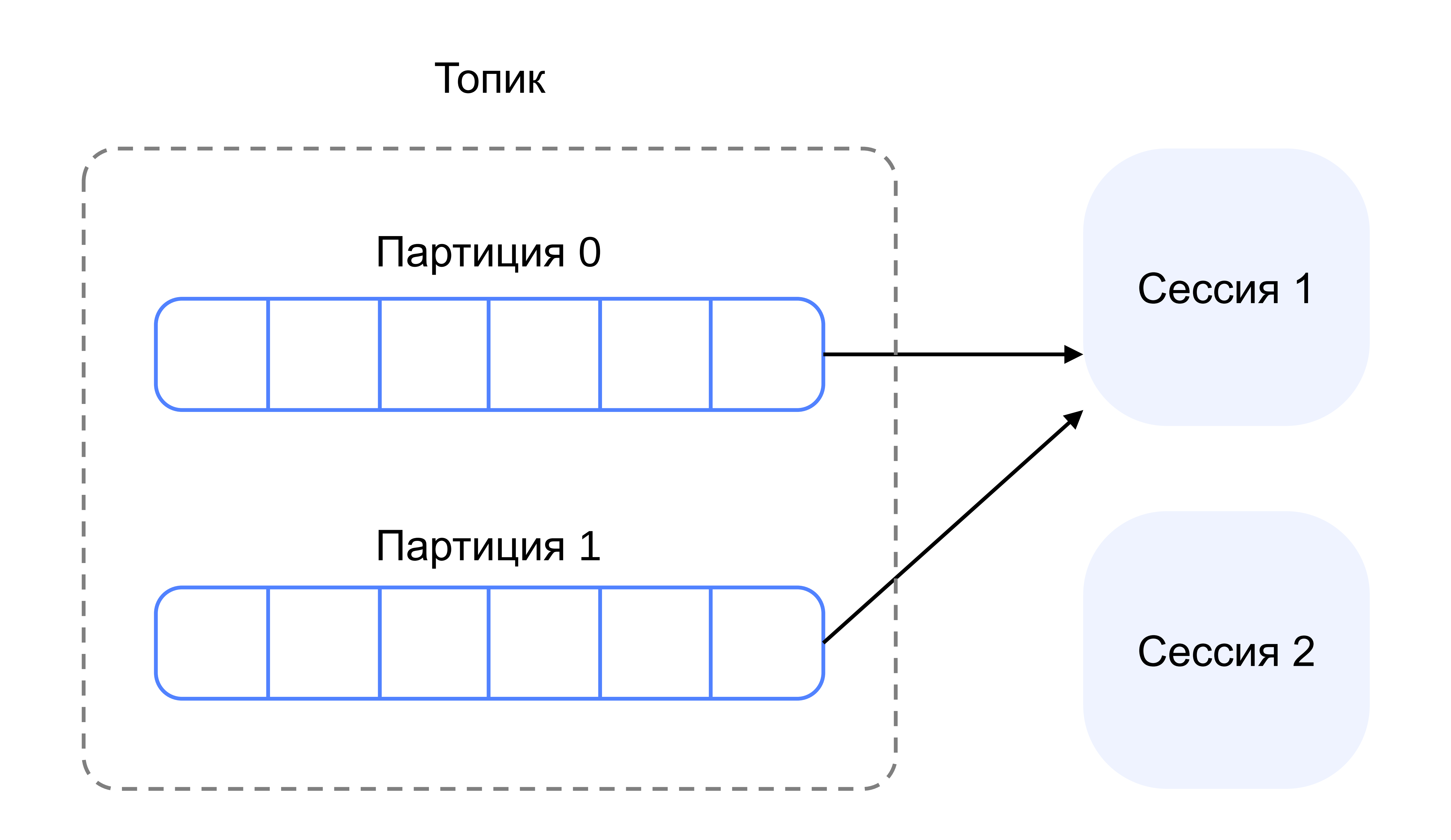
Чтение. Балансировка

- Параллельное чтение топика – несколько сессий с одним consumer_id
- Сервер распределяет партиции динамически

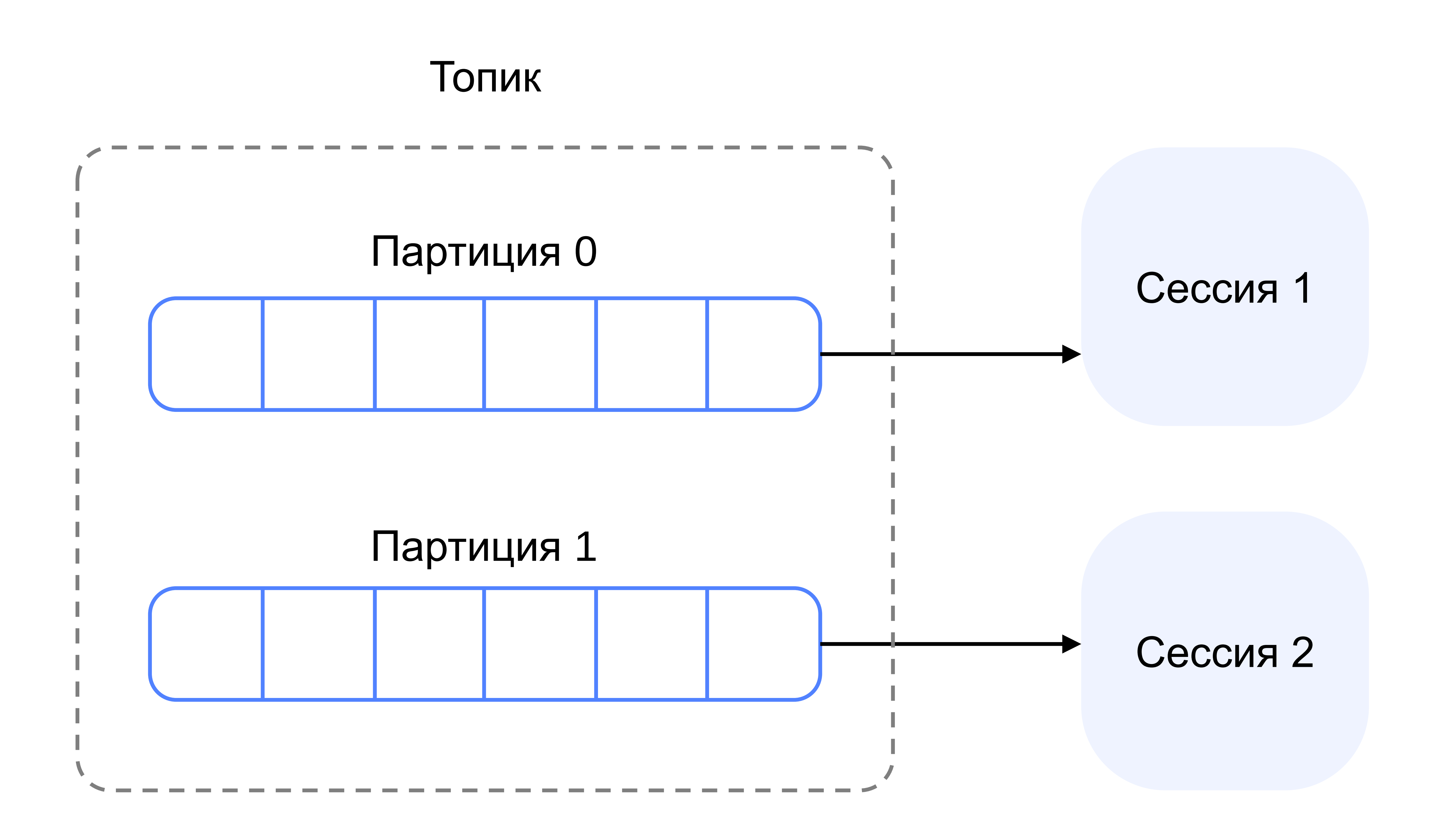
Чтение. Балансировка



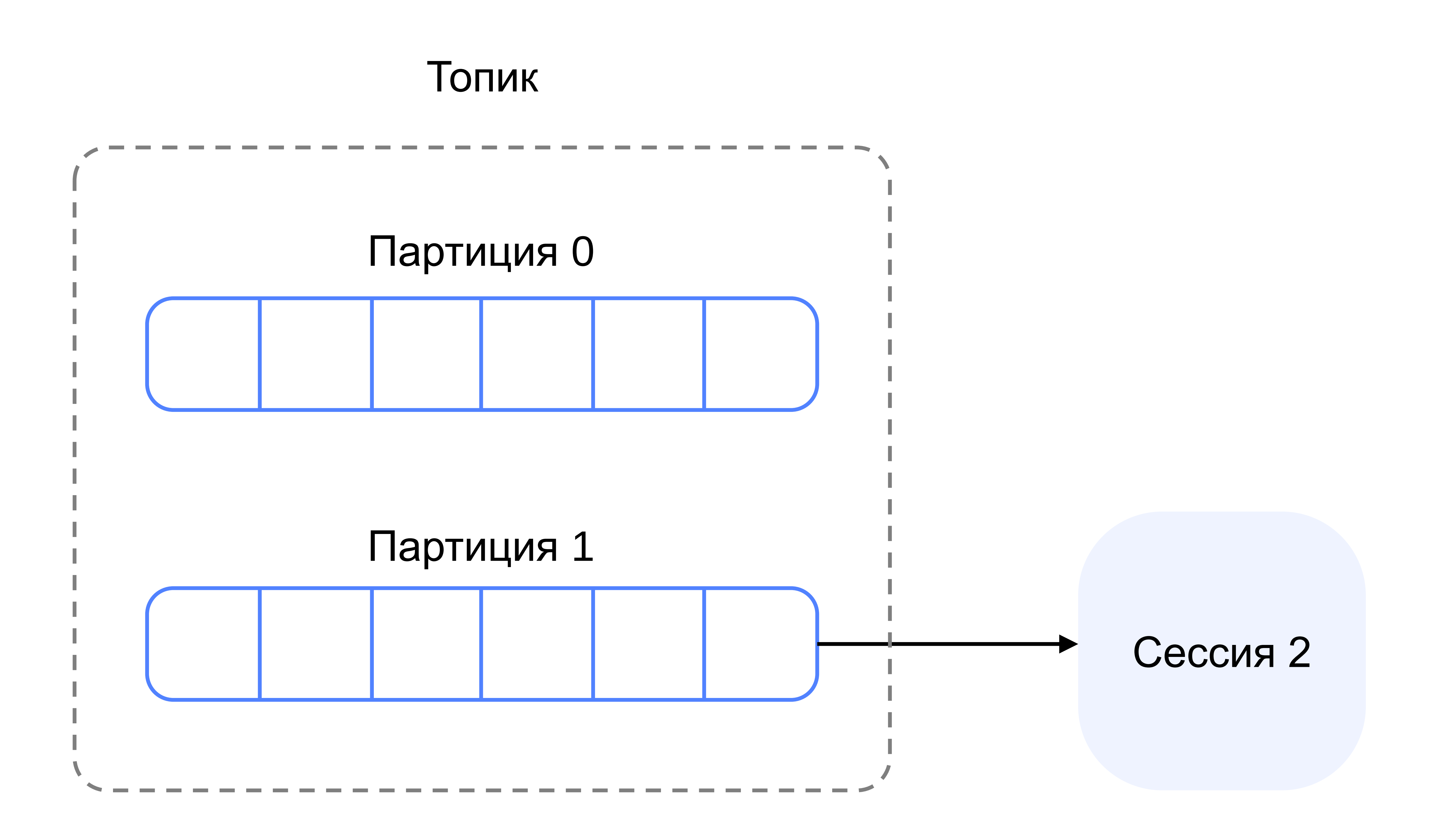
Чтение. Балансировка



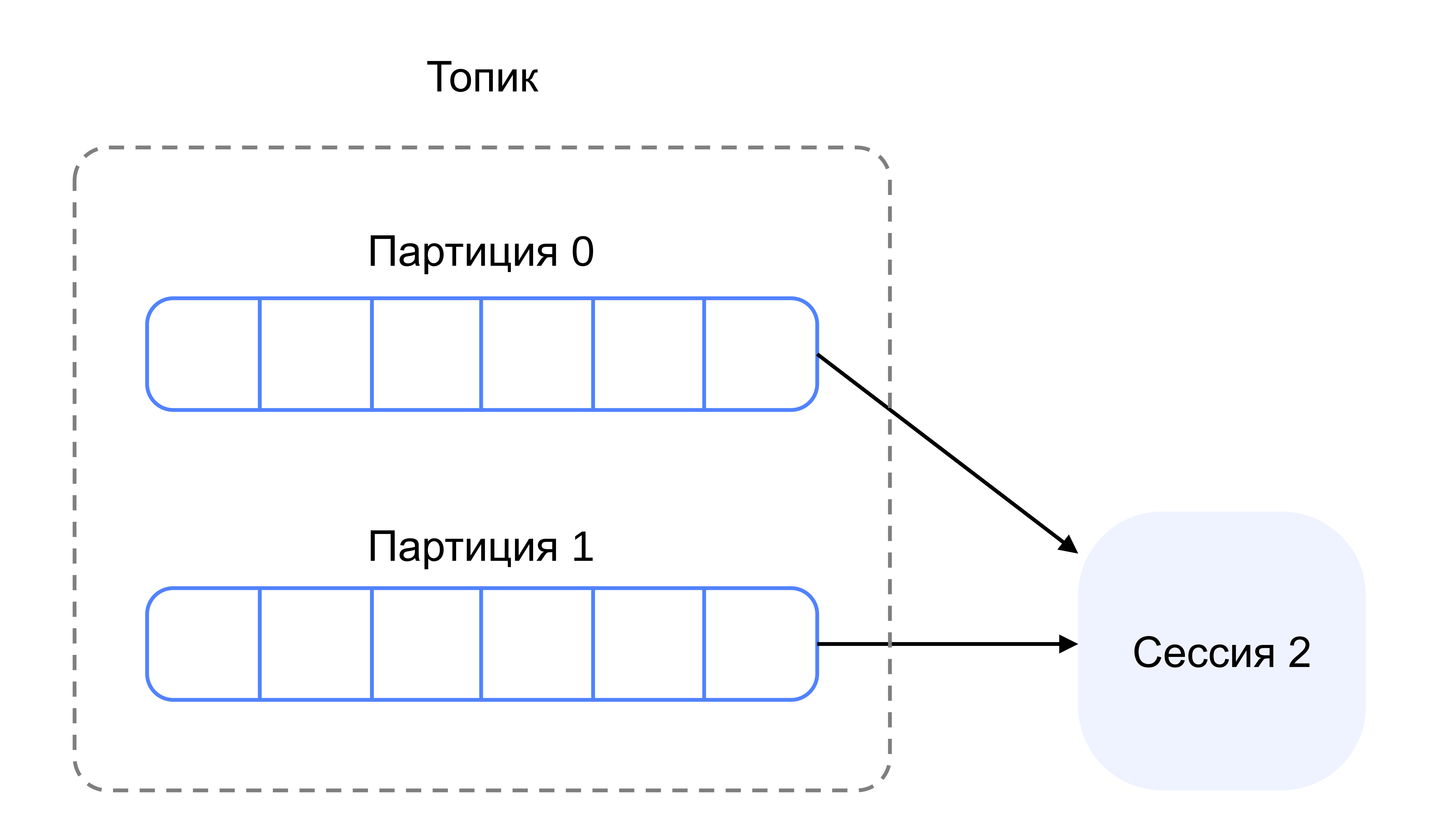
Чтение. Балансировка



Чтение. Балансировка



Чтение. Балансировка



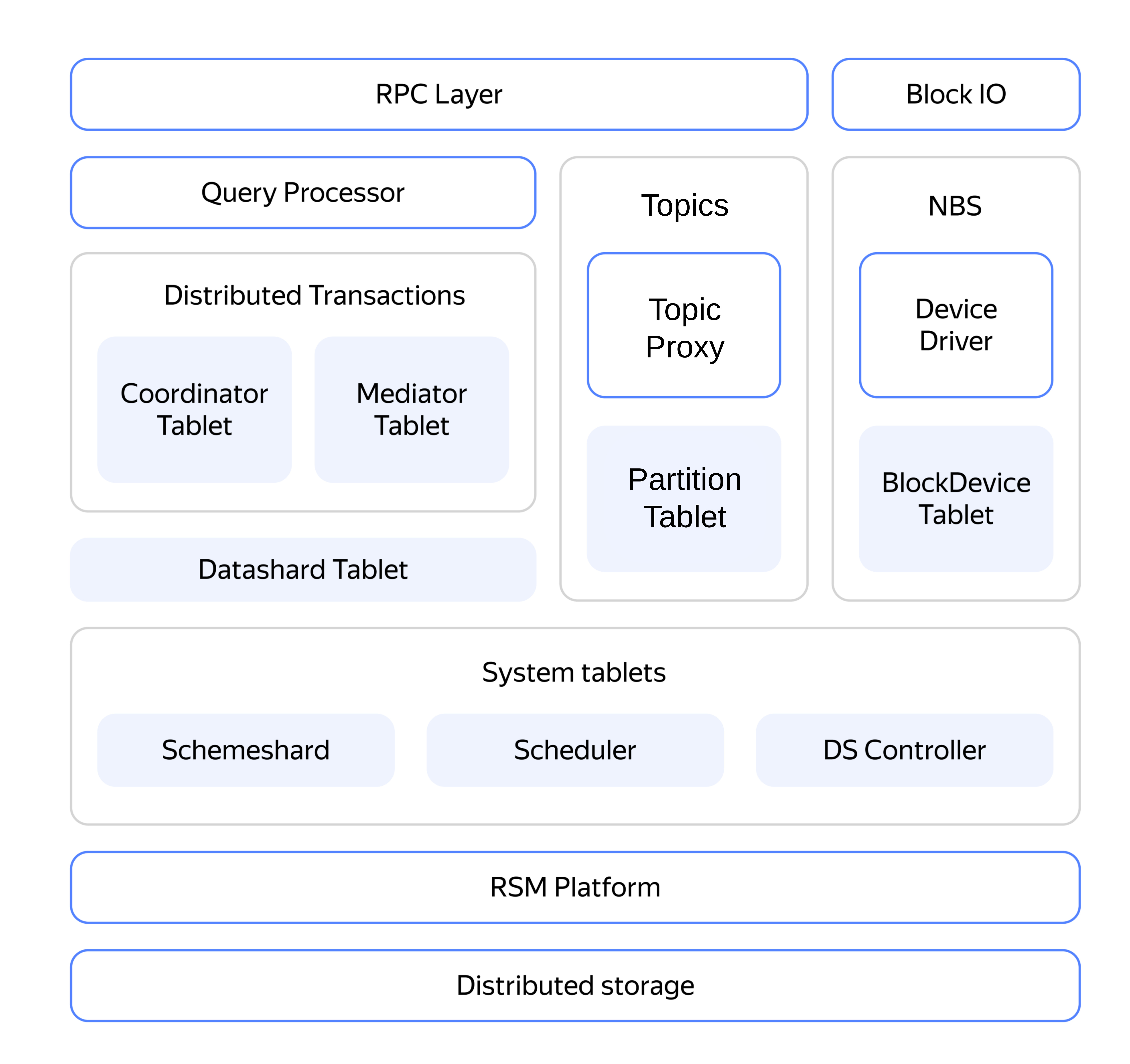
1. Мотивация
2. Модель очереди на основе лога
3. Архитектура решения
4. Использование YDB Topics

Архитектура решения

- Архитектура платформы YDB
- Partition tablet
- Компоненты узла

Платформа YDB

- Таблетка — Replicated state machine
- Уровень хранилища отделён
- Есть разные специализированные таблетки, код наследуется
- Автоматическая перебалансировка таблеток по нодам
- Акторная система как среда коммуникации



Partition Tablet

YDB Tablet предоставляет:

- Интерфейс базы данных
- Локальные транзакции

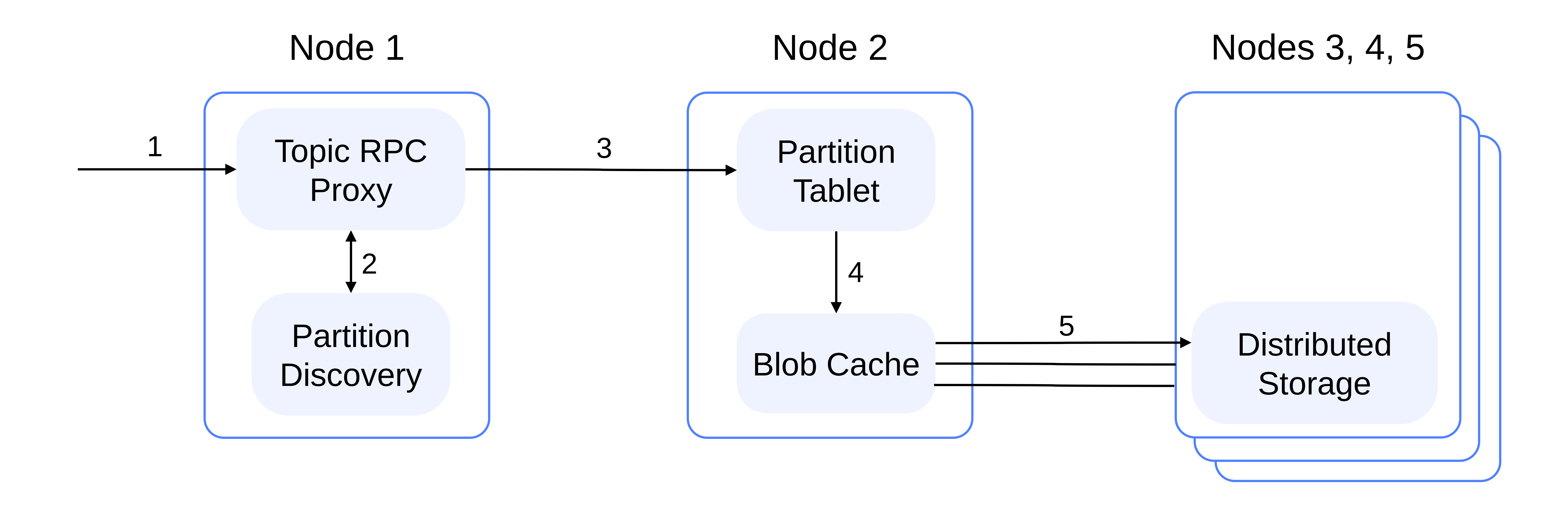
Строим поверх:

- Key-Value API: KV Tablet
- Append-only partition log

Компоненты сервера

- Topic RPC Proxy
- Partition Tablet
- Distributed Storage
- Blob Cache
- Partition Discovery
- Read Balancer Tablet

Partition Tablet. Запись



Балансировка при чтении

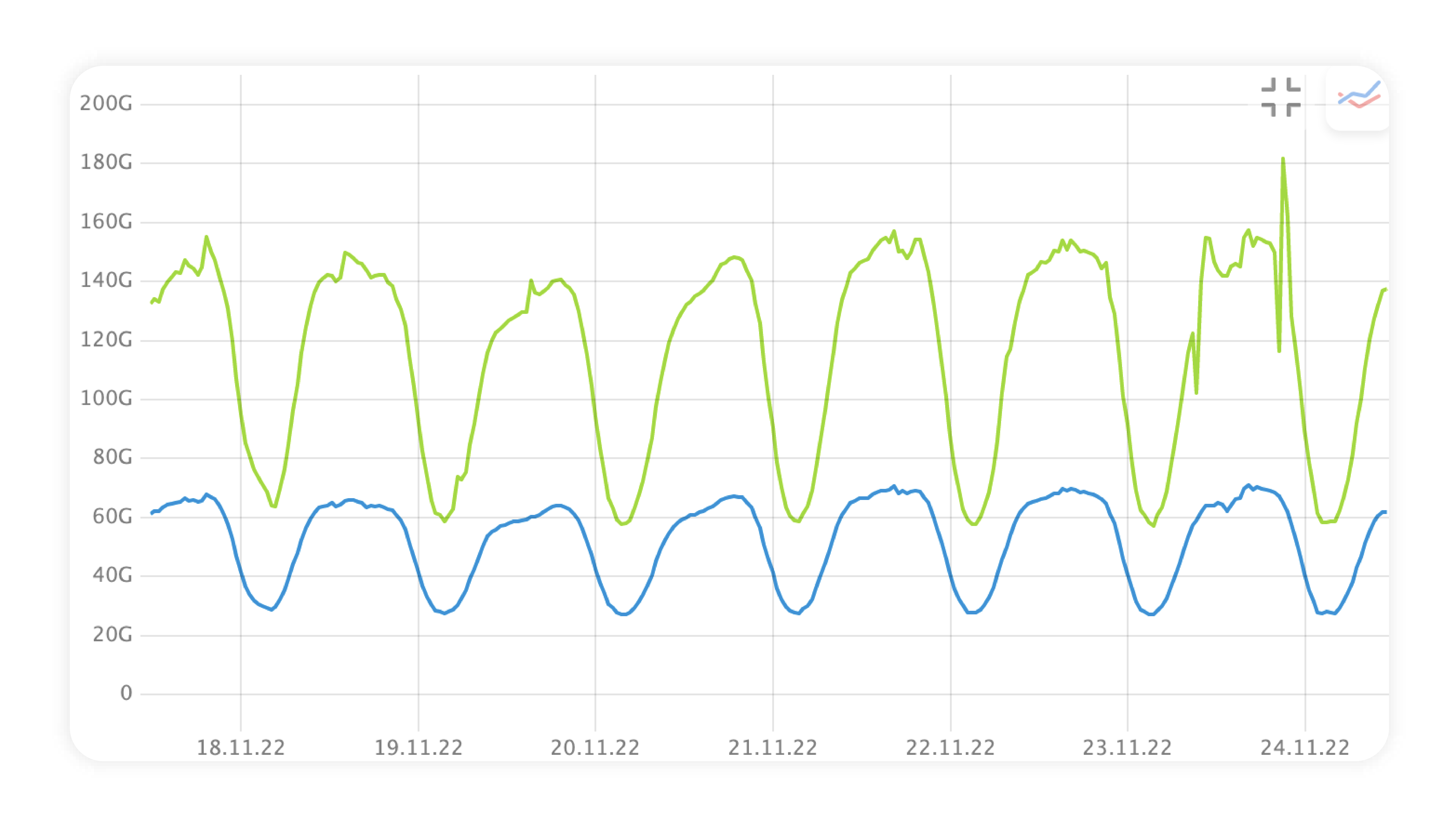
Topic Read Balancer — специальная таблетка

- В ней регистрируются все сессии чтения в данный топик
- Выдаёт и отбирает партиции у сессий

1. Мотивация
2. Модель очереди на основе лога
3. Архитектура решения
4. Использование YDB Topics

Инсталляция YDB Topics в Яндексе

Суммарный трафик записи и чтения за последнюю неделю



- В кластере более 1000 хостов
- Более 1000 аккаунтов пользователей
- ~350 K партиций в ~10 K топиках
- Запись до 70 Гбайт/с
- Чтение до 150 Гбайт/с

SDK и клиенты

- C++ SDK
- Go SDK
- Работа через YDB CLI
- Дополнительно: API AWS Kinesis Datastreams

Итоги

1

Реализация распространённого дизайна очередей на платформе YDB

2

7 лет в проде Яндекса как основной фреймворк распространения данных

3

Open source: экспериментируйте, внедряйте, развивайте

Спасибо!

Ильдар Хисамбеев,

разработчик систем поставки данных, Яндекс

t.me/ildar_khisambeev
t.me/ydb_ru



Яндекс

Обратная связь

